REMARKS

Claims 11-34 are pending, with claims 11, 16, 21, 25 and 33 being independent. Claims 1-10 were previously canceled without prejudice. Reconsideration and allowance of the above-referenced application are respectfully requested.

Claims 11, 12, 15-17, and 20 stand rejected under 35 U.S.C. 102(b) as allegedly being anticipated by Rajwar (Speculative Lock Elision). Claims 11, 15, 16, and 20 stand rejected under 35 U.S.C. 103(a) as allegedly being unpatentable over Transactional Memory (Moss). Claims 1, 6, 7, 14 and 19 stand rejected under 35 U.S.C. 103(a) as allegedly being unpatentable over Moss in view of Lam (Enhancing Software Reliability with Speculative Threads). Claims 21-25, 27 and 31-34 stand rejected under 35 U.S.C. 103(a) as allegedly being unpatentable over Moss in view of Lam in view of Rajwar. Claims 2-5 and 8-10 stand rejected under 35 U.S.C. 103(a) as allegedly being unpatentable over Moss/Lam in view of Christie (U.S. 6,009,512). Claims 13 and 18 stand rejected under 35 U.S.C. 103(a) as allegedly being unpatentable over Rajwar in view of "common prior art". Claim 28 stands rejected under 35 U.S.C. 103(a) as allegedly being unpatentable over Moss/Lam/Rajwar in view of "common prior art". Claim 26 stands rejected under 35 U.S.C. 103(a) as allegedly being unpatentable over Moss/Lam/Rajwar in view of Rajwar.

Claims 29 and 30 stand rejected under 35 U.S.C. 103(a) as allegedly being unpatentable over Moss/Lam/Rajwar in view of "common prior art". These contentions are respectfully traversed.

Paragraphs 3-50 of the 03/07/2007 Office Action are identical to paragraphs 5-52 of 09/21/2006 Office Action. Thus, the Response filed 12/21/2006 is hereby incorporated by reference.

In addition, the Response to Arguments section of the 03/07/2007 Office Action addresses the claim language, "to speculatively read-modify-write a lock variable associated with a critical section", and how this language should be interpreted. The Office Action states:

> Applicant believes that this the amendment requires a greater limitation. Applicant believes that to "speculatively read-modify-write a lock variable" requires that a processor choose <u>to</u> read-modify-write a lock variable rather than choose <u>not to</u> read-modify-write a lock variable.
>
> Again, Examiner disagrees. Perhaps the most common source of speculative execution involves branch instructions. Branch instructions, in most cases, give two choices when they are encountered: 1) the program counter may branch to an address specified within the instruction and 2) the branch instruction is ignored and skipped—allowing the program counter to move to the next consecutive instruction. Branch instructions are very commonly speculatively executed, using a variety of branch prediction techniques.

> One of the simplest branch prediction techniques is called "assume not taken". In this case, the branch instruction is always elided as though it doesn't exist, moving on to the next consecutive instruction. If and when a misprediction (incorrect speculation) occurs, instructions are re-executed to account for the error. Yet, despite the fact that these instructions are always elided when first encountered, it is still said that these branch instructions are "speculative executed".

(*See* 03/07/2007 Office Action at pages 21-22.)

Branch prediction and speculative execution are related, but clearly distinct concepts in processor design. In general, branch prediction facilitates keeping an instruction pipeline full for a processor by using a special fetch/decode unit to predict the direction and outcome of the instructions being executed through multiple levels of branches. By predicting the instruction outcome in advance, the instructions can be executed without waiting on execution of the branch instruction.

Speculative execution involves executing instructions in advance of the actual program counter. In general, a processor's dispatch/execute unit uses dataflow analysis to execute available instructions in the instruction pool and store the results in temporary registers. A retirement unit then searches the instruction pool for completed instructions that are no longer data dependent on other instructions to run, or which have unresolved branch predictions. If any such completed instructions are found, the results are committed to memory by the retirement unit in the order they were originally issued. They are then retired from the pool.

When the "assume not taken" branch prediction technique is employed, the processor assumes that a branch instruction will not cause the program flow to change (i.e., assumes the program counter continues its linear progression through the instructions). Thus, the processor proceeds to fetch instructions that come immediately after the branch instruction, and the processor speculatively executes these fetched instructions until the branch instruction itself is finally executed.

However, the branch instruction itself cannot be considered to be "elided" (i.e., omitted) in this case. The "assume not taken" branch prediction technique refers to the manner in which the fetch/decode unit predicts the branch, not the manner in which the dispatch/execute unit handles the branch instruction. The branch instruction is still dispatched to the instruction pool for execution, and is in fact executed, even if done so speculatively.

For example, two branch instructions in sequence in a program may each be assumed to be not taken, and dispatched to an instruction pool. The second branch instruction may then be speculatively executed if the data on which it depends becomes available before that of the first branch instruction. Regardless of whether or not that second branch instruction is later committed to memory, which will be determined by the outcome of the execution of the first branch instruction, the second branch instruction has nonetheless been speculatively <u>executed</u>.

In view of the above clarifications, it should be apparent that the recited claim language, "to speculatively read-modify-write a lock variable associated with a critical section", <u>does require</u> that one or more instructions that read-modify-write a lock variable are in fact <u>executed</u>, albeit speculatively. This is in stark contrast with Rajwar, which teaches removing the synchronization locks around critical sections entirely. (*See* Rajwar at Section 1, page 295, col. 1; emphasis added.) Rajwar's method does not speculatively read-modify-write a lock variable because Rajwar's method is based on <u>not doing</u> a lock variable write when doing speculation. In other words, choosing "<u>not to</u> read-modify-write a lock variable", as described in Rajwar, cannot be considered equivalent to "speculatively read-modify-write a lock variable", as presently claimed.

Similar arguments apply to Moss, which teaches lock-free synchronization. <u>Lock-free</u> synchronization and using processor speculation in a data processing machine to speculatively read-modify-write a lock variable associated with a critical section are mutually exclusive of each other, by definition. For synchronization to be lock-free, there must be no lock. Moss makes very clear that what is described in Moss is an alternative to using a locking technique. (*See* Moss at p. 289, col. 2.)

Neither Christie nor Lam cure the deficiencies of Rajwar and Moss. Thus, for all of the above reasons, and the reasons given in the Response filed 12/21/2006, independent claims 11, 16, 21, 25 and 33 should be patentable over the cited art. Dependent claims 12-15, 17-20, 22-24, 26-32, and 34 should be patentable over the cited art for at least the above reasons.

Conclusion

It is believed that all of the pending claims have been addressed. However, the absence of a reply to a specific issue or comment does not signify agreement with or concession of that issue or comment. Because the arguments made above may not be exhaustive, there may be reasons for patentability of any or all pending claims (or other claims) that have not been expressed. Finally, nothing in this paper should be construed as an intent to concede any issue with regard to any claim, except as specifically stated in this paper.

It is respectfully suggested for all of these reasons, that the current rejections are overcome, that none of the cited art teaches or suggests the features which are claimed, and therefore that all of these claims should be in condition for allowance. A formal notice of allowance is thus respectfully requested.

No fees are believed to be due with this response. Please apply any necessary charges or credits to deposit account 06-1050.

Respectfully submitted,

Date: May 7, 2007

[illegible]
for Scott C. Harris
Reg. No. 32,030
Attorney for Intel Corporation

WILLIAM E. HUNTER
REG. NO 47,671

Fish & Richardson P.C.
PTO Customer No. 20985
Telephone: (858) 678-5070
Facsimile: (858) 678-5099

10723948.doc